

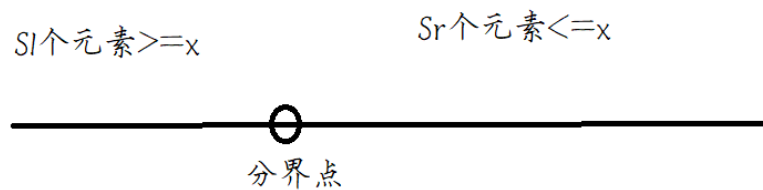
## 6.52

6.52 给出一个分治算法，在一个具有  $n$  个数的数组中找出第二个最大元素。给出你算法的时间复杂性。

```
#include<iostream>
using namespace std;
const int N=1e5+10;
int q[N];
int n,k;
const int IDX=2;
int select(int q[],int l,int r,int k)
{
    if(l>=r) return q[l];
    int i=l-1,j=r+1,x=q[l+r>>1];
    while(i<j)
    {
        do i++; while (x<q[i]);
        do j--; while (x>q[j]);
        if(i<j) swap(q[i],q[j]);
    }
    int s1=j-1+1;
    if(k<=s1) return select(q,l,j,k);
    else return select(q,j+1,r,k-s1);
}
int main()
{
    cin>>n;
    for(int i=0;i<n;i++)
        cin>>q[i];
    cout<<"第二个最大值元素为:"<<select(q,0,n-1,IDX)<<endl;
    for(int i=0;i<n;i++)
        cout<<q[i]<<" ";
    return 0;
}
```

### 算法图解

## 基于快排算法——找第K大的数



设Sl为左区间元素个数，Sr为右区间元素个数

```
if (k $\leq$ sl) select (q, l, j, k) //找到了第K大的数  
else select (q, j+1, r, k-sl)
```

按照降序排序 找到了第K大的数

### 时间复杂度分析

第一次需要遍历 整个数组  $n$

**第二次数组只会递归到左边/右边所以**

第二次平均需要遍历一半的数组  $n/2$

**同理**

第三次  $n/4$

....

最后运行的时间复杂度为  $n(1 + 1/2 + 1/4 + 1/8 + \dots) \leq 2n$

所以时间复杂度是  $O(N)$

### 运行结果

```
10  
50 43 63 97 30 89 89 94 30 33  
第二个最大值元素为:94  
97 94 89 89 63 50 43 33 30 30  
-----  
Process exited after 1.596 seconds with return value 0  
请按任意键继续. . .
```

## 7.5

7.5 用算法 LCS 来找出两个字符串  $A = \text{"xzyzzzyx"}$  和  $B = \text{"zxyyzxz"}$  的最长公共子序列的长度。给出一个最长公共子序列。

### LCS 算法

利用动态规划算法 找到状态方程

观察结论 7.1 如果  $i$  和  $j$  都大于 0, 那么

- 若  $a_i = b_j, L[i, j] = L[i - 1, j - 1] + 1$ ;
- 若  $a_i \neq b_j, L[i, j] = \max\{L[i, j - 1], L[i - 1, j]\}$ 。

下面计算  $A$  和  $B$  的最长公共子序列长度的递推式。可以从观察结论 7.1 立即得出

$$L[i, j] = \begin{cases} 0 & \text{若 } i = 0 \text{ 或 } j = 0 \\ L[i - 1, j - 1] + 1 & \text{若 } i > 0, j > 0 \text{ 和 } a_i = b_j \\ \max\{L[i, j - 1], L[i - 1, j]\} & \text{若 } i > 0, j > 0 \text{ 和 } a_i \neq b_j \end{cases}$$

$$f[i][j] = \max(f[i][j - 1], f[i - 1][j]) \text{ if } (A[i] \neq B[j]) \quad f[i][j] = \max(f[i - 1][j - 1] + 1, f[i][j])$$

```
int LCS(char *a, char *b, int n, int m)
{
    for(int i=0; i<=n; i++)
        f[i][0]=0;
    for(int j=0; j<=m; j++)
        f[0][j]=0;
    for(int i=1; i<=n; i++)
    {
        for(int j=1; j<=m; j++)
        {
            f[i][j]=max(f[i-1][j], f[i][j-1]);
            if(a[i]==b[j]) f[i][j]=max(f[i-1][j-1]+1, f[i][j]);
        }
    }
    return f[n][m];
}
```

### 完整代码

```
#include<iostream>
using namespace std;
const int N=1100;
int f[N][N];
char A[N], B[N];
int n, m;
int LCS(char *a, char *b, int n, int m)
{
    for(int i=0; i<=n; i++)
```

```

        f[i][0]=0;
        for(int j=0;j<=n;j++)
            f[0][j]=0;
        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=m;j++)
            {
                f[i][j]=max(f[i-1][j],f[i][j-1]);
                if(a[i]==b[j]) f[i][j]=max(f[i-1][j-1]+1,f[i][j]);
            }
        }
        return f[n][m];
    }
    int main()
    {
        cin>>n>>m;
        cin>>A+1>>B+1;
        cout<<LCS(A+1,B+1,n,m);
        return 0;
    }

```

## 7.6

7.6 请说明如何修改算法 LCS，使它输出最长公共子序列。

### 输出最长的LCS

结合以下的图表

我们可以在转移的时候记录下，当前状态是由之前的哪一个状态转移而来，最后输出的时候就从后往前递归处理

用  $(-1, 0, 1)$  来表示由哪个状态转移而来

		$j$	0	1	2	3	4	5	6
		$y_j$		<b>B</b>	D	<b>C</b>	A	<b>B</b>	<b>A</b>
$i$	$x_i$								
0		0	0	0	0	0	0	0	0
1	A	0	↑	↑	↑	↖1	←1	↖1	
2	<b>B</b>	0	↖1	↑	←1	↑	↖2	←2	
3	<b>C</b>	0	↑	↑	↖2	←2	↑	↑	
4	<b>B</b>	0	↖1	↑	↑	↑	↖3	←3	
5	D	0	↑	↖2	↑	↑	↑	↑	
6	<b>A</b>	0	↑	↑	↑	↖3	↑	↖4	
7	<b>B</b>	0	↖1	↑	↑	↑	↖4	↑	4

在LCS算法的基础上修改输出算法

```
void print(int n,int m)
{
    if(!n||!m) return ;
    if(state[n][m]==1)
    {
        print(n-1,m-1);
        cout<<a[n];
    }
    else if(state[n][m]==-1) print(n-1,m);
    else print(n,m-1);
}
```

完整代码

```
#include<iostream>
using namespace std;
const int N=1100;
int f[N][N];
int state[N][N];
int n,m;
char a[N],b[N];
void print(int n,int m)
{
    if(!n||!m) return ;
    if(state[n][m]==1)
    {
        print(n-1,m-1);
        cout<<a[n];
    }
}
```

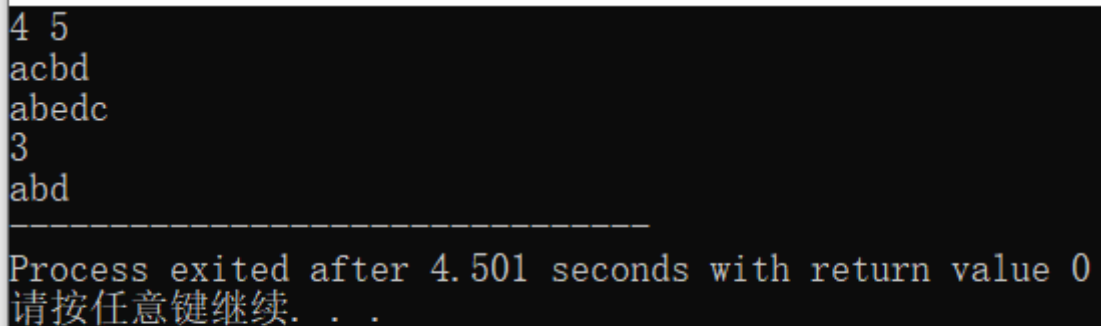
```

        else if(state[n][m]==-1) print(n-1,m);
        else print(n,m-1);
    }
    int main()
    {
        cin>>n>>m;
        cin>>a+1>>b+1;
        for(int j=0;j<=n;j++)
            f[0][j]=0;
        for(int i=0;i<=n;i++)
            f[i][0]=0;

        for(int i=1;i<=n;i++)
        {
            for(int j=1;j<=m;j++)
            {
                if(f[i][j]<f[i-1][j])
                {
                    f[i][j]=f[i-1][j];
                    state[i][j]=-1;
                }
                if(f[i][j]<f[i][j-1])
                {
                    f[i][j]=f[i][j-1];
                    state[i][j]=0;
                }
                if(a[i]==b[j])
                {
                    f[i][j]=max(f[i][j],f[i-1][j-1]+1);
                    state[i][j]=1;
                }
            }
        }
        cout<<f[n][m]<<endl;
        print(n,m);
    }

```

## 运行截图



```

4 5
acbd
abedc
3
abd
-----
Process exited after 4.501 seconds with return value 0
请按任意键继续. . .

```

